

Docket No. 219053US2SRD

**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**

IN RE APPLICATION OF: Mikito IWAMASA

GAU:

SERIAL NO: New Application

EXAMINER:

FILED: Herewith

FOR: METHOD AND COMPUTER PROGRAM PRODUCT FOR SYSTEM DESIGN SUPPORT



**REQUEST FOR PRIORITY**

ASSISTANT COMMISSIONER FOR PATENTS  
WASHINGTON, D.C. 20231

SIR:

- ☐ Full benefit of the filing date of U.S. Application Serial Number , filed , is claimed pursuant to the provisions of 35 U.S.C. §120.
- ☐ Full benefit of the filing date of U.S. Provisional Application Serial Number , filed , is claimed pursuant to the provisions of 35 U.S.C. §119(e).
- ☒ Applicants claim any right to priority from any earlier filed applications to which they may be entitled pursuant to the provisions of 35 U.S.C. §119, as noted below.

In the matter of the above-identified application for patent, notice is hereby given that the applicants claim as priority:

COUNTRY

APPLICATION NUMBER

MONTH/DAY/YEAR

JAPAN

2001-024491

January 31, 2001

Certified copies of the corresponding Convention Application(s)

- ☒ are submitted herewith
- ☐ will be submitted prior to payment of the Final Fee
- ☐ were filed in prior application Serial No. filed
- ☐ were submitted to the International Bureau in PCT Application Number  
Receipt of the certified copies by the International Bureau in a timely manner under PCT Rule 17.1(a) has been acknowledged as evidenced by the attached PCT/IB/304.
- ☐ (A) Application Serial No.(s) were filed in prior application Serial No. filed ; and
- ☐ (B) Application Serial No.(s)
- ☐ are submitted herewith
- ☐ will be submitted prior to payment of the Final Fee

Respectfully Submitted,

OBLON, SPIVAK, McCLELLAND,  
MAIER & NEUSTADT, P.C.

Marvin J. Spivak

Registration No. 24,913

James D. Hamilton  
Registration No. 28,421



22850

Tel. (703) 413-3000  
Fax. (703) 413-2220  
(OSMMN 10/98)

日 本 国 特 許 庁  
JAPAN PATENT OFFICE



別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出 願 年 月 日

Date of Application:

2001年 1月31日

出 願 番 号

Application Number:

特願2001-024491

出 願 人

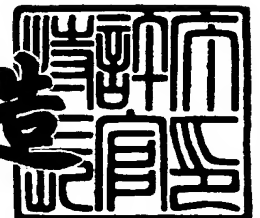
Applicant(s):

株式会社東芝

2001年12月14日

特 許 庁 長 官  
Commissioner,  
Japan Patent Office

及 川 耕 造



出証番号 出証特2001-3108895

【書類名】 特許願

【整理番号】 A000100060

【提出日】 平成13年 1月31日

【あて先】 特許庁長官 殿

【国際特許分類】 G06F 15/00

【発明の名称】 設計支援装置及び設計支援方法並びにシステム設計支援装置

【請求項の数】 18

【発明者】

【住所又は居所】 神奈川県川崎市幸区小向東芝町 1 番地 株式会社東芝研究開発センター内

【氏名】 岩政 幹人

【特許出願人】

【識別番号】 000003078

【氏名又は名称】 株式会社 東芝

【代理人】

【識別番号】 100058479

【弁理士】

【氏名又は名称】 鈴江 武彦

【電話番号】 03-3502-3181

【選任した代理人】

【識別番号】 100084618

【弁理士】

【氏名又は名称】 村松 貞男

【選任した代理人】

【識別番号】 100068814

【弁理士】

【氏名又は名称】 坪井 淳

【選任した代理人】

【識別番号】 100092196

【弁理士】

【氏名又は名称】 橋本 良郎

【選任した代理人】

【識別番号】 100091351

【弁理士】

【氏名又は名称】 河野 哲

【選任した代理人】

【識別番号】 100088683

【弁理士】

【氏名又は名称】 中村 誠

【選任した代理人】

【識別番号】 100070437

【弁理士】

【氏名又は名称】 河井 将次

【手数料の表示】

【予納台帳番号】 011567

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 設計支援装置及び設計支援方法並びにシステム設計支援装置

【特許請求の範囲】

【請求項 1】

システムの状態遷移に関する情報に対し、関連を持つシステムの実行制御に関する情報を付加することによって作成されたシステム仕様に基づいて、システム記述言語で記述された実行可能な形式を持つシステム仕様を生成するシステム生成手段を備えたことを特徴とする設計支援装置。

【請求項 2】

前記状態遷移に関する情報は、少なくとも、システムが取る現状態を示す情報、状態遷移を惹起する契機となるイベントを示す情報、状態遷移したときにシステムが取る次状態を示す情報、及び当該状態遷移に関連する前記実行制御に関する情報を含むものであることを特徴とする請求項 1 に記載の設計支援装置。

【請求項 3】

前記状態遷移に関する情報は、状態が遷移できるための条件を示す情報と、次状態に遷移する前に実行すべきアクションを示す情報との少なくとも一方をさらに含むものであることを特徴とする請求項 2 に記載の設計支援装置。

【請求項 4】

前記実行制御に関する情報は、前記状態遷移に関連して実行制御が惹起される実行処理内容を示す情報及び該実行処理内容に対する実行制御の種類を示す情報を含み、

前記実行制御の種類は、少なくとも、処理の起動、イベント発生に起因する割込による処理の強制終了、イベント発生に起因する割込による処理の一時停止、処理が終了したことを知らせるイベントの通知、割込による処理の一時停止からの復帰を含むことを特徴とする請求項 1 に記載の設計支援装置。

【請求項 5】

前記システム生成手段は、前記システムの状態遷移に関する情報に対し、関連を持つシステムの実行制御に関する情報を付加することによって作成されたシス

テム仕様に基づいて、システムの実行状態の遷移に関する情報を含む実行制御表形式のシステム仕様を生成する実行制御表変換手段を含むことを特徴とする請求項 1 ないし 4 のいずれか 1 項に記載の設計支援装置。

【請求項 6】

前記実行状態の遷移に関する情報は、少なくとも、発生した遷移を示す情報、実行中の実行処理内容を示す情報、実行中の実行処理内容に対する実行制御の種類を示す情報、次に実行すべき実行処理内容、次に実行すべき実行処理内容に対する実行制御の種類を示す情報を含むものであることを特徴とする請求項 5 に記載の設計支援装置。

【請求項 7】

前記システム生成手段は、前記実行制御表形式のシステム仕様に基づいて、システム記述言語で記述された実行可能な形式を持つシステム仕様を生成するシステム実装変換手段を更に含むことを特徴とする請求項 5 または 6 に記載の設計支援装置。

【請求項 8】

前記システム実装変換手段は、個々の前記実行状態の遷移に関する情報を所定の変換ルールに基づいてシステム記述言語で記述された仕様に展開する手段と、展開された個々のシステム記述言語で記述された仕様の所定の仕様統合ルールに基づいて統合する手段とを含むことを特徴とする請求項 7 に記載の設計支援装置。

【請求項 9】

生成された前記システム記述言語で記述されたシステム仕様を実行するための手段を更に備えたことを特徴とする請求項 1 ないし 8 のいずれか 1 項に記載の設計支援装置。

【請求項 10】

システムの状態遷移に関する情報に対し、関連を持つシステムの実行制御に関する情報を付加することによって作成されたシステム仕様に基づいて、システムの実行状態の遷移に関する情報を含む実行制御表形式のシステム仕様を生成し、生成された前記実行制御表形式のシステム仕様に基づいて、システム記述言語

で記述された実行可能な形式を持つシステム仕様を生成することを特徴とする設計支援方法。

#### 【請求項 1 1】

前記状態遷移表形式で記述されたシステム仕様は、少なくとも、システムが取る現状態を示す情報、状態遷移を惹起する契機となるイベントを示す情報、状態遷移したときにシステムが取る次状態を示す情報、及び当該状態遷移に関連する前記実行制御に関する情報を含む状態遷移単位情報の集合からなるものであることを特徴とする請求項 1 0 に記載の設計支援方法。

#### 【請求項 1 2】

前記実行制御に関する情報は、前記状態遷移に関連して実行制御が惹起される実行処理内容を示す情報及び該実行処理内容に対する実行制御の種類を示す情報を含み、

前記実行制御の種類は、少なくとも、処理の起動、イベント発生に起因する割込による処理の強制終了、イベント発生に起因する割込による処理の一時停止、処理が終了したことを知らせるイベントの通知、割込による処理の一時停止からの復帰を含むことを特徴とする請求項 1 1 に記載の設計支援方法。

#### 【請求項 1 3】

前記実行制御表形式のシステム仕様は、少なくとも、発生した遷移を示す情報、実行中の実行処理内容を示す情報、実行中の実行処理内容に対する実行制御の種類を示す情報、次に実行すべき実行処理内容、次に実行すべき実行処理内容に対する実行制御の種類を示す情報を含むプログラム状態遷移単位情報の集合からなるものであることを特徴とする請求項 1 1 または 1 2 に記載の設計支援方法。

#### 【請求項 1 4】

前記実行制御表形式のシステム仕様を生成するに際しては、前記状態遷移単位情報の集合に基づいて前記プログラム状態遷移単位情報の集合を生成し、

前記システム記述言語で記述されたシステム仕様を生成するに際しては、個々の前記プログラム状態遷移単位情報に関する情報を所定の変換ルールに基づいてシステム記述言語で記述された仕様に展開し、展開された個々のシステム記述言語で記述された仕様を所定の仕様統合ルールに基づいて統合することを特徴とす

る請求項 1 3 に記載の設計支援方法。

【請求項 1 5】

設計支援装置としてコンピュータを機能させるためのプログラムを記録したコンピュータ読取り可能な記録媒体であって、

システムの状態遷移に関する情報に対し、関連を持つシステムの実行制御に関する情報を付加することによって作成されたシステム仕様に基づいて、システムの実行状態の遷移に関する情報を含む実行制御表形式のシステム仕様を生成する機能と、

生成された前記実行制御表形式のシステム仕様に基づいて、システム記述言語で記述された実行可能な形式を持つシステム仕様を生成する機能とをコンピュータに実現させるためのプログラムを記録したコンピュータ読取り可能な記録媒体

【請求項 1 6】

設計支援装置としてコンピュータを機能させるためのプログラムであって、

システムの状態遷移に関する情報に対し、関連を持つシステムの実行制御に関する情報を付加することによって作成されたシステム仕様に基づいて、システムの実行状態の遷移に関する情報を含む実行制御表形式のシステム仕様を生成する機能と、

生成された前記実行制御表形式のシステム仕様に基づいて、システム記述言語で記述された実行可能な形式を持つシステム仕様を生成する機能とをコンピュータに実現させるためのプログラム。

【請求項 1 7】

計算に関する仕様および通信に関する仕様を含むシステム仕様モデルを設計するための仕様モデル記述手段と、

特定のアーキテクチャに対しシステム仕様モデルの内容を保ったまま当該システム仕様モデルの持つ構造を分配して、当該システム仕様モデルに含まれる計算に関する仕様および通信に関する仕様の所定部分を当該特定のアーキテクチャに割り当てするためのアーキテクチャ探索手段と、

特定のアーキテクチャに対し仕様要素が割り当てられたシステム仕様モデルに



つき、当該割り当てられた特定のアーキテクチャ上の仕様要素間の通信手順を合成するためのコミュニケーション合成手段と、

システム仕様モデルをもとにハードウェア仕様モデルを生成するためのハードウェア仕様生成手段と、

システム仕様モデルをもとにソフトウェア仕様モデルを生成するためのソフトウェア仕様生成手段とを備え、

前記仕様モデル記述手段は、請求項1ないし9のいずれか1項に記載の設計支援装置に相当する機能を含むものであることを特徴とするシステム設計支援装置

#### 【請求項18】

前記仕様モデル記述手段、前記アーキテクチャ探索手段または前記コミュニケーション合成手段の少なくとも一つにおいて、システム仕様モデルを部品化し、これを設計に再利用するための部品化・再利用手段を更に備えたことを特徴とする請求項17に記載のシステム設計支援装置。

#### 【発明の詳細な説明】

##### 【0001】

#### 【発明の属する技術分野】

本発明は、計算機や電子情報機器における、ハードウェア、ソフトウェア、及びこれら両方を含むシステムの設計支援を行うための設計支援装置及び設計支援方法並びにシステム設計支援装置に関する。

##### 【0002】

#### 【従来の技術】

計算機システムにおいては、UML (Unified Modeling Language) 等のオブジェクト指向手法に代表されるように、システム分析の手法として状態遷移形式が用いられてきた（なお、UMLは例えば「UMLが仆のツク」, Hans-Eriksson/Magnus Penker 著, 杉本宜男/落合修/武田多美子 監訳, ISBN4-8101-8987-2に詳しい）。状態遷移形式は、システムに含まれる複数の状態を抽出し、それら状態が、イベントや条件によって、どのように互いに遷移するかを整理したものである。そして、システム分析が終わった後に、状態遷移形式を参

考にしながら、システム設計を開始するという手順を踏んでいた。

【0003】

一方、システム開発期間の短縮、および設計品質の向上という観点からは、分析をしながら仕様を実際に実行して確かめる、いわゆるラピッドプロトタイピングの機能や、システム分析とシステム設計のシームレスな結合機能が必要不可欠である。

【0004】

従来の技術では、以上の事情に鑑み、状態遷移による分析結果に基づくシミュレーションを実行したり、実行プログラムに変換する手法が実現されてきた (Better State)。これらの技術では、状態遷移の瞬間に実行されるアクション、及びシステムが状態に滞留するときに実行されるアクティビティに、実行プログラムを割り当てることにより、シミュレーション機能や変換機能を実現している。

【0005】

【発明が解決しようとする課題】

状態遷移形式では、アクションは瞬時に実行が終了することが前提となっており、0より大きい一定の実行時間を必要とする計算機プログラムに代表される実行処理内容を割り当てることは、意味論的に不整合であり、システム全体の挙動の整合性が保証できない。

【0006】

また、アクティビティは状態と処理が一对一に対応しているので、遷移が異なれば遷移先や遷移元が同じ状態でも異なるアクションが実行されるという状態遷移形式の記述の柔軟性が損なわれ、記述できる範囲が限定されてしまうという問題がある。この結果、状態遷移形式のアクションやアクティビティに計算機プログラム等の実行処理内容を割り当てる方式では、精度のよいシミュレーションが実行できなかったり、また柔軟なシステム設計ができないという課題があった。

【0007】

本発明は、上記事情を考慮してなされたもので、システム分析から設計までを一環して支援し、システム分析結果に対応した精度良いシステム設計の支援を行

って、システム開発期間を短縮させ、設計品質を向上させることの可能な設計支援装置及び設計支援方法並びにシステム設計支援装置を提供することを目的とする。

【0008】

【課題を解決するための手段】

本発明に係る設計支援装置は、システムの状態遷移に関する情報に対し、関連を持つシステムの実行制御に関する情報を付加することによって作成されたシステム仕様に基づいて、システム記述言語で記述された（例えば、計算機や電子部品組込み機器によって）実行可能な形式を持つシステム仕様を生成するシステム生成手段を備えたことを特徴とする。

【0009】

好ましくは、前記状態遷移に関する情報は、少なくとも、システムが取る現状態を示す情報、状態遷移を惹起する契機となるイベントを示す情報、状態遷移したときにシステムが取る次状態を示す情報、及び当該状態遷移に関連する前記実行制御に関する情報を含むものであるようにしてもよい。

【0010】

好ましくは、前記システム生成手段は、前記システムの状態遷移に関する情報に対し、関連を持つシステムの実行制御に関する情報を付加することによって作成されたシステム仕様に基づいて、システムの実行状態の遷移に関する情報を含む実行制御表形式のシステム仕様を生成する実行制御表変換手段を含むようにしてもよい。好ましくは、前記実行状態の遷移に関する情報は、少なくとも、発生した遷移を示す情報、実行中の実行処理内容を示す情報、実行中の実行処理内容に対する実行制御の種類を示す情報、次に実行すべき実行処理内容、次に実行すべき実行処理内容に対する実行制御の種類を示す情報を含むものであるようにしてもよい。好ましくは、前記システム生成手段は、前記実行制御表形式のシステム仕様に基づいて、システム記述言語で記述された実行可能な形式を持つシステム仕様を生成するシステム実装変換手段を更に含むようにしてもよい。

【0011】

本発明に係る設計支援方法は、システムの状態遷移に関する情報に対し、関連

を持つシステムの実行制御に関する情報を付加することによって作成されたシステム仕様に基づいて、システムの実行状態の遷移に関する情報を含む実行制御表形式のシステム仕様を生成し、生成された前記実行制御表形式のシステム仕様に基づいて、システム記述言語で記述された実行可能な形式を持つシステム仕様を生成することを特徴とする。

## 【 0 0 1 2 】

好ましくは、前記状態遷移表形式で記述されたシステム仕様は、少なくとも、システムが取る現状態を示す情報、状態遷移を惹起する契機となるイベントを示す情報、状態遷移したときにシステムが取る次状態を示す情報、及び当該状態遷移に関連する前記実行制御に関する情報を含む状態遷移単位情報の集合からなるものであるようにしてもよい。好ましくは、前記実行制御表形式のシステム仕様は、少なくとも、発生した遷移を示す情報、実行中の実行処理内容を示す情報、実行中の実行処理内容に対する実行制御の種類を示す情報、次に実行すべき実行処理内容、次に実行すべき実行処理内容に対する実行制御の種類を示す情報を含むプログラム状態遷移単位情報の集合からなるものであるようにしてもよい。好ましくは、前記実行制御表形式のシステム仕様を生成するに際しては、前記状態遷移単位情報の集合に基づいて前記プログラム状態遷移単位情報の集合を生成し、前記システム記述言語で記述されたシステム仕様を生成するに際しては、個々の前記プログラム状態遷移単位情報に関する情報を所定の変換ルールに基づいてシステム記述言語で記述された仕様に展開し、展開された個々のシステム記述言語で記述された仕様を所定の仕様統合ルールに基づいて統合するようにしてもよい。

## 【 0 0 1 3 】

また、本発明に係るシステム設計支援装置は、計算に関する仕様および通信に関する仕様を含むシステム仕様モデルを設計するための仕様モデル記述手段と、特定のアーキテクチャに対しシステム仕様モデルの内容を保ったまま当該システム仕様モデルの持つ構造を分配して、当該システム仕様モデルに含まれる計算に関する仕様および通信に関する仕様の所定部分を当該特定のアーキテクチャに割り当てるためのアーキテクチャ探索手段と、特定のアーキテクチャに対し仕様要

素が割り当てられたシステム仕様モデルにつき、当該割り当てられた特定のアーキテクチャ上の仕様要素間の通信手順を合成するためのコミュニケーション合成手段と、システム仕様モデルをもとにハードウェア仕様モデルを生成するためのハードウェア仕様生成手段と、システム仕様モデルをもとにソフトウェア仕様モデルを生成するためのソフトウェア仕様生成手段とを備え、前記仕様モデル記述手段は、請求項 1 ないし 9 のいずれか 1 項に記載の設計支援装置に相当する機能を含むものであることを特徴とするシステム設計支援装置。

## 【 0 0 1 4 】

なお、装置に係る本発明は方法に係る発明としても成立し、方法に係る本発明は装置に係る発明としても成立する。

また、装置または方法に係る本発明は、コンピュータに当該発明に相当する手順を実行させるための（あるいはコンピュータを当該発明に相当する手段として機能させるための、あるいはコンピュータに当該発明に相当する機能を実現させるための）プログラムとしても成立し、該プログラムを記録したコンピュータ読取り可能な記録媒体としても成立する。

## 【 0 0 1 5 】

本発明によれば、状態遷移形式の柔軟さを保持しつつ全体のシステムとしての整合性も保証する方式で、実行処理内容を遷移状態形式に割り当てて、この状態遷移形式から、実行可能なシステムを生成する手段を具備する設計支援装置を提供することができる。

## 【 0 0 1 6 】

そして、本発明によれば、状態遷移表形式に処理内容（プログラム）を直接入力して、これを実行可能なシステムに変換するシステム生成手段を具備しているので、状態遷移表の柔軟さを損なずにシステム設計を行うことができ、また処理内容の実行制御命令自体は瞬間に実行されるので意味論的に状態遷移の仕組みと整合性が取れたシステム設計を行うことができる。

## 【 0 0 1 7 】

従って、本発明によれば、システム分析から設計までを一環して支援し、システム分析結果に対応した精度よいシステム設計の支援を行って、システム開発期

間を短縮させ、設計品質を向上させることができる。

【0018】

【発明の実施の形態】

以下、図面を参照しながら発明の実施の形態を説明する。

【0019】

図11に、本発明の実施の形態に係るシステム設計支援装置100の構成例を示す。図11に示されるように、システム設計支援装置100は、仕様モデル記述部101、アーキテクチャ探索部102、コミュニケーション合成部103、ハードウェア仕様生成部104、部品化・再利用部105、ソフトウェア仕様生成部106、システム仕様記録部107を備えている。システム設計支援装置100は、例えば、コンピュータを用いて構成可能であり、この場合、各処理部分はプログラムによって実施できる。

【0020】

システム仕様記録部107は、入力データや各処理部の処理結果の全部又は一部、例えば、計算機で実行されるソフトウェアの仕様や、半導体を組み合わせたハードウェアの仕様や、ソフトウェアとハードウェアを組み合わせた組み込みシステムの仕様や、ワークフロー等のビジネスプロセスの仕様に対しシステムレベルでの仕様等や、その他の必要なデータを記録するためのものである。

【0021】

仕様モデル記述部101は、設計者が予め定められた仕様記述形式に従って計算内容の仕様、通信内容の仕様を記述することを支援するためのものである。仕様モデル記述部101の支援の結果、仕様モデルが生成される。仕様記述形式には、例えば、構造化プログラミング言語に代表される構造化テキスト形式や、グラフを利用する構造図形式や、テーブルを利用する表形式が有る。

【0022】

アーキテクチャ探索部102は、与えられた仕様記述モデルと、アーキテクチャ（例えば、ハードウェア実行環境とソフトウェア実行環境の構成）に対し、仕様の内容を保存したまま仕様モデルの部分構造を分割しアーキテクチャ要素に分配する。すなわち、仕様モデル記述部101で設計された計算内容の仕様や通信

内容の仕様を構成する部品をアーキテクチャ要素に割り当てる。

【0023】

コミュニケーション合成部103は、アーキテクチャ探索部102で分配された通信仕様要素間に対して通信手順（プロトコル）を挿入し、通信内容の仕様が挿入された通信手順と整合性を保つためのプロトコル変換（通信手順の組替え）を行う。

【0024】

なお、アーキテクチャ探索部102が処理対象とする仕様記述モデルは、仕様モデル記述部101を利用して記述されたものに限定されず、他のツール等で記述されたものでもよい。また、コミュニケーション合成部103が処理対象とする仕様記述モデルは、アーキテクチャ探索部102の出力結果（またはこの結果に修正を加えたもの）に限定されず、他のツール等で記述されたものでもよい。

【0025】

ハードウェア仕様生成部104は、システム仕様からハードウェア記述言語などによるハードウェア仕様を生成する。

【0026】

ソフトウェア仕様生成部106は、システム仕様からソフトウェア記述言語などによるソフトウェア仕様を生成する。

【0027】

ハードウェア仕様生成部104やソフトウェア仕様生成部106は、仕様モデル記述部101の出力結果（またはこの結果に修正を加えたもの）、アーキテクチャ探索部102の出力結果（またはこの結果に修正を加えたもの）、コミュニケーション合成部103の出力結果（またはこの結果に修正を加えたもの）、あるいは他のツール等で記述されたものなどを処理対象とすることができる。

【0028】

部品化・再利用部105は、仕様モデル記述部101、アーキテクチャ探索部102、コミュニケーション合成部103の全部または一部の設計段階において、仕様を部品化し、これを設計に再利用するためのものである。

【0029】

以下詳述する設計支援装置は、図 1 1 のシステム設計支援装置の仕様モデル記述部 1 0 1 に用いられるもので、状態に基づくシステム分析からプログラムのシステム設計をシームレスに実行できることを可能にするものである。

【0 0 3 0】

図 1 に、本発明の一実施形態に係る設計支援装置の構成例を示す。

【0 0 3 1】

図 1 に示されるように、本設計支援装置は、実行制御表変換部 1 1 およびシステム実装変換部 1 2 を含むシステム生成部 1、外部とデータや指示のやり取りを行うための入出力部 2、入力となる状態遷移表形式で記述された仕様（2 1）、中間状態となる実行制御表形式で記述された仕様（2 2）及び出力となるシステム記述言語で記述された仕様（2 3）、並びに変換ルール及び仕様統合ルールなどを記憶するための記憶部（図示せず）を備えている。また、必要に応じて、通信部 3 やプログラム実行部 4 などをも更に備えることができる。

【0 0 3 2】

システム実装変換部 1 2 の実行制御表変換部 1 1 は、状態遷移表形式で記述された仕様（2 1）を、実行制御表形式で記述された仕様（2 2）に変換する。

【0 0 3 3】

システム実装変換部 1 2 のシステム実装変換部 1 2 は、実行制御表形式で記述された仕様（2 2）を、システム記述言語で記述された仕様（2 3）に変換する。

【0 0 3 4】

本設計支援装置は、例えば、コンピュータ上で本発明を実施したプログラム（設計支援ソフト）を実行することによって、実現可能である。

【0 0 3 5】

以下では、まず「状態遷移表形式で記述された仕様（状態遷移表形式の仕様）」と「実行制御表形式で記述された仕様（実行制御表形式の仕様）」の意味内容を説明した上で、前者を後者に変換する実行制御表変換部 1 1 の処理について説明する。

【0 0 3 6】



# <状態遷移表形式の仕様>

遷移状態形式の仕様とは、システムの挙動を、システムが取り得る有限個の状態と「イベント」による状態間の遷移の定義により記述する仕組みである（図4あるいは図7の具体例参照）。

## 【0037】

状態遷移表形式の仕様は、例えば下記に示す状態遷移単位の集合によって構成される。

状態遷移単位＝（遷移番号、現状態、イベント、遷移条件、  
アクション、次状態、実行制御）

なお、状態遷移単位＝（遷移番号、現状態、イベント、遷移条件、次状態、実行制御）、状態遷移単位＝（遷移番号、現状態、イベント、アクション、次状態、実行制御）、状態遷移単位＝（遷移番号、現状態、イベント、次状態、実行制御）なども可能である。

## 【0038】

遷移番号で識別される個々の状態遷移単位は、システムが「現状態」にあるとき、「イベント」が発生して、かつ「遷移条件」を満たせば、「アクション」を実行して「次状態」に遷移する、というシステムの挙動の一部を表している。このとき、遷移時のアクションは一瞬で終了する。「実行制御」は、遷移が発生したときの、実行処理内容（プログラム）の実行制御に関する項目である。

## 【0039】

例えば図4のCDプレーヤの具体例（ただしアクションは省かれている）における番号[2]の状態遷移単位は、再生状態（playing）において、ポーズ（pause）イベントが発生した場合に、playプログラムに割り込み（interrupt）をかけて一時停止させるとともに、pauseプログラムを起動（start）させ、ポーズ（pause）状態に遷移する、という仕様を表している。

## 【0040】

なお、特別な状態として、初期状態（start）、終了状態（end）が準備される。

初期状態 (start) は、システムが起動したときの最初の状態を指示するポインタであり、状態遷移単位の一つとして以下のように指示が表現される。

(start、－、－、－、最初の状態)

終了状態 (end) は、終了したときに遷移する仮想的な状態である。

【0041】

また、状態遷移表形式は、階層的に記述することもできる。すなわち、下位階層の状態遷移単位の集合を一つの階層状態Xにまとめて、上位階層でXを一つの状態として含む状態遷移表形式の仕様を定義することができる。上位階層において階層状態Xに遷移したときは、Xの下位階層では初期状態 (start) で指示する状態に遷移する。

【0042】

<遷移の接続について>

状態遷移表形式では、現状態から次状態への「遷移」を定義するが、当該次状態は、次の時間では、現状態になるので、状態遷移単位同士の間には同一の「状態」を介して、方向性のある隣接関係が成立する。この隣接関係が成立する遷移同士は「接続している」と呼ぶことにする。遷移の接続関係は、遷移の方向性に従う半順序関係である（例：「推移律が成り立つ」＝「遷移Aと遷移Bが接続し、遷移Bと遷移Cが接続するならば、遷移Aと遷移Cは接続関係にある」）。また、遷移の方向に従った遷移接続を順方向の接続、その逆を逆方向の接続と呼ぶことにする。例えば、遷移Aとが遷移Bが接続し、接続の方向が遷移Aから遷移Bである（遷移Aの後で遷移Bが発生する）とき、遷移Aから遷移Bへの接続が順方向になり、遷移Bから遷移Aへの接続が逆方向になる。

【0043】

<「実行制御」の細目>

状態遷移形式における「実行制御」の項目では、プログラムの実行制御に関する細項目を用意して、この細項目に処理内容の単位（プログラム）を入力する。ここで、細項目とは、少なくとも、起動 (start)、停止 (stop)、割込み一時停止 (interrupt)、プログラムの終了 (finish)、一時停止からの復帰 (resume)、及びこれらの要素を組み合わせた項目で構

成される。

【0044】

なお、処理内容の単位（プログラム）が終了したことも、イベントとみなす。これを、特別に、終了イベントと呼び、状態遷移表形式では [FINISH] と表記するものとする（図4の番号[3]の状態遷移単位参照）。

【0045】

＜実行制御表形式の仕様＞

実行制御表形式の仕様とは、システムにおけるプログラム等の実行処理内容の単位と、これらの処理内容単位（プログラム）が状態遷移によってどのように実行制御（起動、停止、割込）されるかを定義することにより、システムの挙動を記述する仕組みである（図8の具体例参照）。

【0046】

実行制御表形式の仕様は、処理内容単位（プログラム）が状態遷移によってどのように切り替わるかを示すプログラム状態遷移単位の集合によって構成される。

【0047】

プログラム状態遷移単位は、状態遷移に起因する処理内容単位（プログラム）の実行状態（待機状態、実行状態、終了状態、一時停止状態）の変化を、他プログラムの実行状態の変化と関連付けたテーブルである。

【0048】

プログラム状態遷移単位は、例えば以下にテーブルで表現される。

プログラム状態遷移単位＝（番号、現プログラム、遷移、移動タイプ、  
終了タイプ、次プログラム）

個々のプログラム状態遷移単位は、「現プログラム」実行中に「遷移」が発生すると、「終了タイプ」に従って「現プログラム」を実行制御し、「起動タイプ」に従って「次プログラム」を起動するという仕様を表している。

【0049】

例えば図8のCDプレーヤの具体例における番号（2）のプログラム状態遷移単位は、現プログラム（play）の実行中に遷移[2]が発生すると、現プロ

グラム (play) を割込一時停止させ、次現プログラム (pause) を起動させる、という仕様を表している。

【0050】

起動タイプには、「起動」、「復帰」の2種類がある。

「起動」は、プログラムを起動することを示す。

「復帰」は、一時停止されていたプログラムの実行を再開することを示す。

終了タイプには、「割込終了」、「完了」、「割込一時停止」の3種類がある。

「完了」は、前プログラムが終了したときに次プログラムを起動することを示す。

「割込終了」は、現プログラムが実行中に、遷移が発生すると、現プログラムを強制終了して、次プログラムを起動することを示す。

「割込一時停止」は、現プログラムが実行中に、遷移が発生すると、現プログラムを一時停止して、次プログラムを起動することを示す。

【0051】

また、例えば、

(現プログラムA、遷移X、移動タイプY、終了タイプZ、次プログラムB)

(現プログラムA、遷移X、移動タイプY、終了タイプZ、次プログラムC)

のように、同一の(現プログラム、遷移X、移動タイプY、終了タイプZ)の組み合わせに対し複数の「次プログラム」が定義されるときには、当該複数の次プログラム(上記の例では、B、C)が遷移Xにより並列起動されることを表すこととする。

【0052】

なお、実行制御表形式の仕様でも、状態遷移表形式の仕様と同様に、特別なプログラム状態として、開始状態(start)と終了状態(end)がある。

【0053】

また、実行制御表形式の仕様でも、階層的に記述することもできる。すなわち、下位階層のプログラム状態遷移単位の集合を一つの階層状態Xにまとめて、上位階層でXを実行処理内容(プログラム)の一つとして含むプログラム状態遷移

表形式の仕様を定義することができる。上位階層において階層状態Xに遷移したときは、Xの下位階層では初期状態（start）で指示されるプログラム状態が実行される。また、階層状態Xに対して「割込終了」「割込一時停止」が行われたときは、Xの下位階層全てのプログラムに対して「割込終了」「割込一時停止」が行われる。

【0054】

<実行制御表変換部11の処理>

続いて、実行制御表変換部11の処理について説明する。

【0055】

実行制御表変換部11は、状態遷移表形式における実行制御の項目（start, stop, interrupt, resume, finish）の情報に基づき、状態遷移表を実行制御表形に変換する。

【0056】

図2に、実行制御表変換部11の処理手順の一例を示す。

【0057】

<<ステップS1：プログラム状態遷移単位への展開>>

状態遷移表における個々の状態遷移単位に対し、実行制御の開始に関する項目（start, resume）から一つプログラムを選択して、これを「現プログラム」とするとともに、実行制御の終了に関する項目（stop, finish, interrupt）から一つプログラムを選択して、これを「次プログラム」として得られる、<現プログラム、次プログラム>の組み合わせを展開する。

【0058】

現プログラムの由来となる実行制御項目（start, resume）に応じて、次のように「起動タイプ」を決定する。

start ならば、「起動タイプ」=起動

resume ならば、「起動タイプ」=復帰

また、次プログラムの由来となる実行制御項目（FINISH, interrupt, stop）に応じて、次のように「終了タイプ」を決定する。

stop ならば、「起動タイプ」=割込終了  
finish ならば、「起動タイプ」=完了  
interrupt ならば、「起動タイプ」=割込一時停止  
また、「遷移」には、遷移番号を割り当てる。

【0059】

1つの状態遷移単位に対して、1つ以上のプログラム状態遷移単位が展開される。

【0060】

以上の処理を状態遷移表に含まれる全ての状態遷移単位に対して行うことにより、プログラム状態遷移単位の集合すなわち実行制御表を得る。

【0061】

<<ステップS2：遷移を検索し、プログラム状態遷移単位を生成する>>

状態遷移表形式を実行制御表形式に変換した時点において、プログラム状態単位、

(現プログラム、遷移、起動タイプ、終了タイプ、次プログラム)

の中に、現プログラムや次プログラムが無いものがある場合は、状態遷移表における遷移の接続関係を参照して、「遷移」に対して、

「次プログラム」項目が無い場合は、順方向、

「現プログラム」項目が無い場合は、逆方向、

となる遷移番号を「遷移」に持つプログラム状態遷移単位を検索し、「次プログラム」、「現プログラム」に相当する処理単位(プログラム)に突き当たるまで、これを繰り返す。

【0062】

プログラムが見つかったときは、見つかったプログラムを各々「次プログラム」、「現プログラム」とする新たなプログラム状態遷移単位を作成し、検索を行った過程のプログラム状態遷移単位(複数)を削除する。このとき、新たに作成したプログラム状態遷移単位の「遷移」は、検索を行った全ての遷移の遷移列とする。

【0063】

### <<ステップ S 3 : 新しい処理内容の追加>>

見つけたプログラムと元々のプログラムとの間を介在する新しい実行処理（プログラム）を追加する。遷移の接続性を考慮して、起動・終了部分に適切に挿入することで実現される。

【0064】

以上の手順に従い、状態遷移表が実行制御表に変換される。

【0065】

### <状態遷移表形式と実行制御形式の違い>

状態遷移表形式は、イベントに応じた状態の変化により、システムの挙動を記述する。システム分析過程では、重要な記述である。システム設計の段階においては、システム分析の結果を踏まえ、今度は状態ではなく、計算機プログラム等の実行処理内容を同定し、どのタイミングで、どの実行処理内容を起動したり停止したりするかに関する制御方式を設計する。

【0066】

状態遷移表形式でも、State Chart 方式に代表されるように、遷移時に実行されるアクションや、状態滞留中に実行されるアクティビティとして、処理内容を状態遷移に記述する方法もあるが、状態遷移形式ではアクションは瞬時に実行が終了することが前提となっており、0 より大きい一定の実行時間を必要とする計算機プログラムに代表される実行処理内容を割り当てることは意味論的に不整合であり、システム全体の挙動の整合性が保証できない。また、アクティビティは状態と処理が一对一に対応しているので、遷移が異なれば遷移先や遷移元が同じ状態でも異なるアクションが実行されるという状態遷移形式の記述の柔軟性が損なわれ、記述できる範囲が限定されてしまうという問題があり、結果として状態遷移形式のアクションやアクティビティに計算機プログラム等の実行処理内容を割り当てる方式では、精度のよいシミュレーションが実行できなかったり、また柔軟なシステム設計ができないという課題がある。

そこで、状態遷移ではなく、処理内容の実行制御を定義する新しい方式が、実行制御表形式である。

【0067】

次に、実行制御表形式で記述された仕様を、システム記述言語で記述された仕様に変換する、システム実装変換部 1 2 について説明する。

【0068】

以下では、まず「システム記述で記述された仕様」の意味内容を説明した上で、「実行制御表形式で記述された仕様」を「システム記述で記述された仕様」に変換するシステム実装変換部 1 2 の処理について説明する。

【0069】

＜システム記述言語＞

システム記述言語とは、計算機や電子部品回路で実現されるシステムにおける処理実行の動作仕様を記述する言語で、

少なくとも、並列処理、割込み処理、同期処理、繰り返し実行処理、逐次実行処理を記述できる手段を有し、さらにこれらの処理要素を用いた階層関係でシステムの仕様を記述することができる記述言語であり、かつ、

適切な実装変換装置により、計算機や電子部品組込み機器の上で実行可能な形式に変換できるものとする。ここで、実装変換装置とは、例えば、計算機のソフトウェアで実現されるシステムの場合は、コンパイラと呼ばれる変換装置に相当する。

【0070】

＜SpecC言語＞

システム記述言語の代表例としては、仕様記述言語 SpecC が挙げられる。なお、SpecC については例えば “SpecC: Specification Language and Methodology” , Daniel D. Gajski, Kluwer Academic Publishers, Dordrecht, ISBN0-7923-7822-9 に詳しい。

SpecC では、par (並列)、fsm (繰り返し+逐次)、try / trap / itrp (割込み)、notify / wait (同期) という特別な言語要素を C / C++ に追加した仕様記述言語である (図 10 の具体例参照)。

【0071】

以下の説明では、システム記述言語によって記述された仕様を、SpecC の言語要素をもとにし、下記に列挙する形式により表現する場合を例にとって説明



する。

【0072】

・ p a r { A , B , C }

この言語要素の例では、A、B、Cを並列実行する。

【0073】

・ f s m { { 1 , A , g o t o ( 2 ) } ,  
          { 2 , B , f l g == 3 : g o t o ( 1 ) , f l g == 1 : g o t o ( 2 ) } ,  
          }

ここで、f s mの各要素は、

{ラベル, 処理内容, {条件: 条件成立時の遷移}, ...}、あるいは、

{ラベル, 処理内容, 処理終了時の遷移, ...}、あるいは、

{ラベル, 処理内容}、あるいは、

処理内容

のいずれかで構成され、特に指定が無い場合は、左から順に逐次実行される。すなわち、遷移ラベルの項目が無い場合、{ラベル, 処理内容}は、処理内容が終了すると、次のf m s要素を実行することになる。また、ラベル=1あるいは最左のf m s要素が最初に実行されるものとする。遷移は、「g o t o ( X ) : Xはラベル番号」にて表す。例えば、g o t o ( 1 ) は、最初のf m s要素に戻ることを表している。

上記の例では、Aが実行され、終了すれば、Bが実行され、Bが終了したとき、変数f l gの値が3ならばAを、1ならばBを実行することを表している。

次に例示するように、ラベルの無い要素は、ラベルを用いた或る実行制御の簡略形であるといえる。

f s m { A , B , C } = f s m { { 1 , A , g o t o ( 2 ) } , { 2 , B , g o t o ( 3 ) } , { 3 , C , ... } , ... }  
・ t r y { A } t r a p ( e v 1 ) { B } i t r p ( e v 2 ) { C } ...

この言語要素の例では、まず、Aを実行する。そして、Aの実行中にイベントe v 1が発生すれば、Aを強制終了して、Bを実行する。また、Aの実行中にイベントe v 2が発生すれば、Aを一時停止して、Cを実行し、Cが終了したとき、Aの処理を再開する。

なお、`trap (e) {X}` , `itrp (e2) {Y}` は、複数あってもよい

【0074】

・ `wait (ev)`

これは、イベント `ev` の発生を待つ同期処理である。

【0075】

・ `notify (ev)`

これは、イベント `ev` を発生させる同期処理である。

【0076】

・ `flg = X`

これは、変数 `flg` への値の代入である。

【0077】

・ `flg == X`

これは、条件判断である。

【0078】

また、階層構造とは、処理内容がさらにシステム記述言語で書かれた詳細な仕様に展開されることを意味することとする。

階層構造を持つ仕様の例を次に示す。

```
par {
    fsm {A, wait (ev2), B}
    try {C} trap (ev1) {notify (ev2)}
}
```

この仕様例の意味を自然言語で書き下ろすと、次のようになる。

「AとCが最初に並列に実行される。イベント `ev1` が着たら、Cを強制終了し、イベント `ev2` を発生する。 `ev2` の発生した時点までにAが終了していれば、これと同期してBを起動する。」

<システム実装変換部12の処理>

続いて、システム実装変換部12の処理について説明する。

【0079】

前述のように、システム実装変換部 1 2 は、実行制御表形式で記述された仕様を、システム記述言語で記述された仕様に変換する。

#### 【0080】

図 3 に、システム実装変換部 1 2 の処理手順の一例を示す。

#### 【0081】

＜＜ステップ S 1 1 : 変換ルールによるシステム記述言語要素への展開＞＞

変換ルールとは、実行制御表におけるプログラム状態遷移単位（1 行）を、システム記述言語による仕様に展開するルールである。

#### 【0082】

システム実装変換部 1 2 は、変換ルールに従って、実行制御表の全てのプログラム状態遷移単位をシステム記述言語の仕様に展開する。

#### 【0083】

変換ルールの一例としては、例えば「次プログラム」の処理内容

（現プログラム A、遷移 T、移動タイプ、終了タイプ、次プログラム B）において、＜起動タイプ、終了タイプ＞の組み合わせに対して、下記に列挙するルールを適用して、システム記述言語で書かれた仕様の部分に展開するものがある。なお、下記において「ev\_\_XX」とあるのは、イベントを表しているものとする。

#### 【0084】

・＜起動、割込終了＞の場合は、

```
fsm {wait (ev_A_stop) , B, goto(1)} と
```

```
try {A} trap {ev_T} {notify (ev_A_stop) } に展開する。
```

・＜起動、完了＞の場合は、

```
fsm {wait (ev_A_end) , B, goto(1)} と
```

```
fsm {A, notify (ev_A_end) , goto(1)} に展開する。
```

・＜起動、割込一時停止＞の場合は、

```
fsm {wait (ev_A_interrupt) , B, goto(1)} と
```

```
try {A} itrp {ev_T} {fsm {n tify (ev_A_interrupt) , wait (ev_A_re  
sume) } } に展開する。
```

- ・ <復帰、完了>の場合は、

fsm {wait (ev\_A\_end) , notify (ev\_B\_resume) , goto(1)} と  
fsm {A, notify (ev\_A\_end) , goto(1)} に展開する。

- ・ <復帰、割込終了>の場合は、

fsm {wait (ev\_A\_stop\_resume) , notify (ev\_B\_resume) , goto(1)} と

try {A} itrp {ev\_T} {notify (ev\_A\_stop\_resume) } に展開する。

【0085】

システム実装変換部12は、これらのような展開を、実行制御表に含まれる全てのプログラム状態遷移単位に対して実行する。

【0086】

<<ステップS12：処理内容単位の仕様としてまとめる>>

展開された仕様の一部の集合を、その個々の処理内容（A，B，…）を中心に  
して次の形式にまとめる。

f m s {開始待ち、処理内容、完了後処理、遷移}、あるいは、

f m s {開始待ち、処理内容、遷移}

ここで、開始待ちについては、

開始待ち=par {wait (同期イベント1) , wait (同期イベント2) }、

あるいは、

開始待ち=wait (同期イベント1, 同期イベント2)

の形式をとるものとする。

また、処理内容については、

処理内容=処理内容A，B，…、あるいは、

処理内容=try {A} trap (遷移イベント) {遷移後処理} と

try {A} itrp (遷移イベント) {遷移後処理} とから

構成されるリスト

の形式をとるものとする。

また、完了後処理については、

完了後処理=n o t i f y (同期イベント)

の形式をとるものとする。

また、遷移については、

遷移 = goto (X), Xはラベル

の形式をとるものとする。

#### 【0087】

ここで説明のために変換ルールにて展開された仕様記述を以下のように分類する。

##### (1) 開始の fsm

開始の fsm とは、処理内容 B に対して、

fsm {wait (X), B, goto (1)}

の形式を持つ要素を呼ぶこととする。

#### 【0088】

##### (2) 終了の try

終了の try とは、処理内容 X に対して、

try {X} trap (e) {Y}

try {X} itrp (e) {Y}

の形式を持つ要素を呼ぶこととする。

##### (3) 終了の fsm

終了の fsm とは、処理内容 X に対して、

fsm {X, notify (e), goto (1)}

の形式を持つ要素を呼ぶこととする。

#### 【0089】

以下に、仕様統合のルールの例を列挙する。

・『開始の fsm と終了の fsm の統合』

fsm {wait (e), A, goto (1)}

fsm {A, notify (f), goto (1)}

を、

fsm {wait (e), A, notify (f), goto (1)}

として統合する。

・『開始の fsm 同士の統合』

fsm {wait (A), B, goto (1)}

fsm {wait (C), B, goto (1)}

を、

fsm {wait (A, C), B, goto (1)}

として統合する。

開始に wait が複数ある場合には、これの論理和（少なくとも1つのイベントを持つ）に統合する。

【0090】

・『終了の try 同士の統合』

try {A} trap (e1) {C}

try {A} trap (e2) {D}

try {A} trap (e3) {E}

を、

try {A} trap (e1) {C} trap (e2) {D} trap (e3) {E}

に統合する。

・『開始の fsm と終了の try の統合』

fsm {wait (A), B, goto (1)}

try {B} trap (e) {C}

を、

fsm {wait (A), try {B} trap (e) {C}, goto (1)}

に統合する。同様に、

fsm {wait (A), B, goto (1)}

try {B} itrp (e) {C}

を、

fsm {wait (A), try {B} itrp (e) {C}, goto (1)}

に統合する。

・『終了の fsm と終了の try の統合』

```
fsm {A, notify (end), goto (1)}
try {B} trap (e) {C}
```

を、

```
fsm {try {fsm {A, notify (FINISH)}}
      trap (e) {C}
      trap (FINISH) {notify (end)},
      goto (1)}
```

に統合する。ここで、FINISH は、A の終了を知らせるイベントであり、これを trap (FINISH) にて検知して、最終的に A の終了イベント end を発行する。

【0091】

<<ステップ S13 : 全体のシステム仕様としてまとまる>>

ステップ S12 によって得られた仕様の一部を、並列実行構造 (par) にて束ねると、最終的なシステム仕様が完成する。

システム仕様＝

```
par {
    fsm {開始待ち, 処理内容 A, 完了後処理},
    fsm {開始待ち, 処理内容 B, 完了後処理},
    fsm {開始待ち, 処理内容 C, 完了後処理},
    :
}
```

以上の手順により、実行制御表形式で記述された仕様がシステム記述言語で記述された仕様に変換される。

【0092】

なお、この手順で用いた変換ルールは一例であり、生成されるシステム記述言語の最適化（例えば記述の大きさを最小にする）や実際の計算や電子機器による実装の容易性のために、他の変換ルールを用いることもできる。

## 【0093】

また、得られたシステム仕様についても、機能を保存したまま、変換を行い最適化を行う等の手段を用いて、所望の構造に変換することもできる。

## 【0094】

さて、以下では、本実施形態について、Compact Diskの演奏装置(CD player)の設計を例として、さらに詳しく説明する。ここでのCD playerとは、例えば、制御ボタン(制御ボタンはハードウェアのボタンでもGUI画面上のボタンでもよい)として、play(演奏)、pause(一時停止)、stop(停止)を押して、演奏機械を制御する装置である。なお、以下では、<XXX>は、制御ボタンXXXが押されたイベントを意味するものとする。

## 【0095】

ここで、CD playerは、その状態として、stopped(停止中)、playing(演奏中)、pausing(一時停止中)の3つの状態を持つものとし、制御ボタンの押し下げをイベントとする状態遷移表形式でCD playerの仕様を記述する。図4に、この場合の状態遷移表の一例を示す。

## 【0096】

状態遷移表の各行は、現状態に対してイベントが発生したとき、どの状態に遷移するかを記述している。また、実行制御の項目の細目に、start, stop, interrupt, resumeがあり、それぞれ、遷移時に、起動(start)、割込終了(stop)、割込一時停止(interrupt)、割込一時停止からの復帰(resume)を行う、処理内容の単位(プログラム)が定義してある。CD playerにおける処理内容の単位は、play(演奏実行)、pause(一時停止実行)の2種類である。

## 【0097】

また、終了(finish)は、実行中の処理内容が終了したことをイベントとして利用することを示し、例えば、遷移番号[3]では、playが終了したことがplayingからstoppedに遷移するイベントであると定義している。このとき、イベント欄には、[FINISH]と記述していて、終了のイ



イベントであることを明示している。

【0098】

また、start 状態からの遷移（遷移 [0]）の場合は、開始プログラム状態（start）を強制終了するものとみなす。

【0099】

＜実行制御への変換＞

実行制御表変換部 11 により状態遷移表を実行制御表に変換する。

【0100】

まず、図 2 のステップ S1 の手順に従い、状態遷移表の実行制御項目において、開始に関する項目（start, resume）と終了に関する項目（stop, finish, interrupt）両方に処理内容（プログラム）がある状態遷移単位を選択し、これを＜現プログラム、次プログラム＞の組み合わせとして展開する。

【0101】

開始に関する項目と終了に関する項目とを両方持っているものは、遷移 [2]，[4]，[5] であるから、最初に展開されるプログラム状態遷移単位は、図 5 のようになる。

【0102】

次に、図 2 のステップ S2 の手順に従い、遷移の接続関係を辿ると、「次プログラム」が無い遷移 [7]，[3]，[6]，[0] に対しては、それぞれ遷移の接続の順方向を参照する。例えば遷移 [7] の場合は、stopped 状態を介して遷移 [1] と順方向に接続しており、遷移 [1] には、「次プログラム」があるので、図 6 に示す新たなプログラム状態遷移を得る。遷移 [3]，[6]，[0] の場合も同様に、図 6 に示す新たなプログラム遷移状態を得る。

【0103】

ここで、図 2 のステップ S3 の手順に従い、遷移 [0]，[3]，[6]，[7] と [1] の間に新しい処理内容（Xstop）を挿入することにする。

【0104】

開始と終了を考慮して、

状態遷移表における遷移 [1] の s t a r t 項目

状態遷移表における遷移 [7], [3], [6], [0] の s t o p 項目  
に挿入する。

【0105】

図4の状態遷移表にX s t o p が挿入されたもの（最終的に得られる実行制御表に対応する状態遷移表）を図7に、最終的に得られる実行制御表を図8に示す。

【0106】

<システム仕様への変換>

システム実装変換部12により実行制御表形式の仕様をシステム記述言語形式の仕様に変換する。

【0107】

まず、図3のステップS11の手順と、変換ルールに従い、各々のプログラム状態遷移単位をS p e c C形式に変換する。この結果を、図9に示す。

【0108】

これら処理内容 ( p l a y , p a u s e , X s t o p ) 毎にまとめると、次のようになる。

【0109】

[ p l a y ]

```
fsm {wait (ev_Xstop_stop) , play, goto(1)}
fsm {play, notify (ev_play_end) , goto(1)}
try {play} itrp (ev_[2]) {fsm {notify (ev_play_interrupt) ,
                                wait (ev_play_resume) } }
try {play} trap (ev_[6]) {notify (ev_play_stop) }
```

[ p a u s e ]

```
fsm {wait (ev_play_interrupt) , pause, goto(1)}
try {pause} trap (ev_[5]) {notify (ev_pause_st p) }
try {pause} trap (ev_[7]) {notify (ev_pause_st p) }
try {pause} trap (ev_[4]) {n tify (ev_pause_st p_resume) }
```

[X s t o p]

```
fsm {wait (start) , Xstop, g to(1)}
fsm {wait (ev_play_end) , Xstop, goto(1)}
fsm {wait (ev_play_stop) , Xstop, goto(1)}
fsm {wait (ev_pause_stop) , Xstop, goto(1)}
try {Xstop} trap (ev_[1]) {notify (ev_Xstop_stop) }
```

[O T H E R S]

```
fsm {wait (ev_pause_stop_resume) , notify (ev_play_resume) , goto(
1))
fsm {wait (ev_pause_stop_resume) , notify (ev_play_resume) , goto(
1))
```

次に、図3のステップS12の手順と仕様統合ルールに従って、統合を行う。  
この結果を、以下に示す。

【0 1 1 0】

[X s t o p]

```
fsm {wait (start) , Xstop, goto (1) }
fsm {wait (ev_play_end) , Xstop, goto (1) }
fsm {wait (ev_play_stop) , Xstop, goto (1) }
fsm {wait (ev_pause_stop) , Xstop, goto (1) }
try {Xstop} trap (ev_ [1] ) {notify (ev_Xstop_stop) }
```

は、開始の f s m を統合して、

```
fsm {wait (start, ev_play_end, ev_play_stop, ev_pause_stop) , Xsto
p, goto (1) }
try {Xstop} trap (ev_ [1] ) {notify (ev_Xstop_stop) }
```

とし、開始の f s m と終了の t r y を統合して、

```
fsm {wait (start, ev_play_end, ev_play_stop, ev_pause_stop) ,
    try {Xstop} trap (ev_ [1] ) {notify (ev_Xstop_stop) } ,
    got (1)
}
```

とする。

【0 1 1 1】

[p l a y]

fsm {wait (ev\_Xstop\_stop) , play, goto (1) }

fsm {play, notify (ev\_play\_end) , goto (1) }

try {play} (ev\_ [2] ) itrp {fsm {notify (ev\_play\_intrrupt) ,  
wait (ev\_play\_resume) } }

try {play} (ev\_ [6] ) trap {notify (ev\_play\_stop) }

は、終了の t r y を統合し終了の f s m を t r y に統合すると、

fsm {wait (ev\_Xstop\_stop) , play, goto (1) }

fsm {

try {fsm {play, notify (FINISH) } }

itrp (ev\_ [2] ) {fsm {notify (ev\_play\_intrrupt) , wait (e  
v\_play\_resume) } }

trap (ev\_ [6] ) {notify (ev\_play\_stop) }

trap (FINISH) {notify (ev\_play\_end) } ,

goto (1)

}

とする。さらに、開始の f s m と終了の f s m を統合して、

fsm {wait (ev\_Xstop\_stop) ,

try {fsm {play, notify (FINISH) } }

itrp (ev\_ [2] ) {fsm {notify (ev\_play\_intrrupt) , wait (e  
v\_play\_resume) } }

trap (ev\_ [6] ) {notify (ev\_play\_stop) }

trap (FINISH) {notify (ev\_play\_end) } ,

goto (1)

}

とする。

【0 1 1 2】

[p a u s e]

```
fsm {wait (ev_play_interrupt) , pause, goto (1) }
try {pause} trap (ev_ [5] ) {notify (ev_pause_stop_resume) }
try {pause} trap (ev_ [7] ) {notify (ev_pause_stop) }
try {pause} trap (ev_ [4] ) {notify (ev_pause_stop_resume) }
```

は、t r y を統合して、

```
fsm {wait (ev_play_interrupt) , pause, goto (1) }
try {pause} trap (ev_ [5] ) {notify (ev_pause_stop_resume) }
      trap (ev_ [7] ) {notify (ev_pause_stop) }
      trap (ev_ [4] ) {notify (ev_pause_stop_resume) }
```

とし、さらに起動の f s m と終了の t r y を統合して、

```
fsm {wait (ev_play_interrupt) ,
      try {pause} trap (ev_ [5] ) {notify (ev_pause_stop_resume) }
      trap (ev_ [7] ) {notify (ev_pause_stop) }
      trap (ev_ [4] ) {notify (ev_pause_stop_resume) } ,
      goto (1)
}
```

とする。

【0 1 1 3】

[O T H E R S]

同じものは1つに統合して、

```
fsm {wait (ev_pause_stop_resume) , notify (ev_play_resume) , goto
(1) }
```

とする。

【0 1 1 4】

最終的に得られる、システム記述言語形式の仕様は、図 1 0 のようになる。

【0 1 1 5】

なお、以上の各機能は、ソフトウェアとして実現可能である。

また、本実施形態は、コンピュータに所定の手段を実行させるための（あるい

はコンピュータを所定の手段として機能させるための、あるいはコンピュータに所定の機能を実現させるための) プログラムとして実施することもでき、該プログラムを記録したコンピュータ読取り可能な記録媒体として実施することもできる。

【0116】

なお、この発明の実施の形態で例示した構成は一例であって、それ以外の構成を排除する趣旨のものではなく、例示した構成の一部を他のもので置き換えたり、例示した構成の一部を省いたり、例示した構成に別の機能あるいは要素を付加したり、それらを組み合わせたりすることなどによって得られる別の構成も可能である。また、例示した構成と論理的に等価な別の構成、例示した構成と論理的に等価な部分を含む別の構成、例示した構成の要部と論理的に等価な別の構成なども可能である。また、例示した構成と同一もしくは類似の目的を達成する別の構成、例示した構成と同一もしくは類似の効果を奏する別の構成なども可能である。

また、この発明の実施の形態で例示した各種構成部分についての各種バリエーションは、適宜組み合わせて実施することが可能である。

また、この発明の実施の形態は、個別装置としての発明、関連を持つ2以上の装置についての発明、システム全体としての発明、個別装置内部の構成部分についての発明、またはそれらに対応する方法の発明等、種々の観点、段階、概念またはカテゴリに係る発明を包含・内在するものである。

従って、この発明の実施の形態に開示した内容からは、例示した構成に限定されることなく発明を抽出することができるものである。

【0117】

本発明は、上述した実施の形態に限定されるものではなく、その技術的範囲において種々変形して実施することができる。

【0118】

【発明の効果】

本発明によれば、状態遷移表形式に処理内容(プログラム)を直接入力して、これを実行可能なシステムに変換するシステム生成手段を具備しているので、状

態遷移表の柔軟さを損なずにシステム設計を行うことができ、また処理内容の実行制御命令自体は瞬間に実行されるので意味論的に状態遷移の仕組みと整合性が取れたシステム設計を行うことができる。これによって、システム分析から設計までを一環して支援し、システム分析結果に対応した精度よいシステム設計の支援を行って、システム開発期間を短縮させ、設計品質を向上させることができる。

【図面の簡単な説明】

【図 1】

本発明の実施の形態に係る設計支援装置の構成例を示す図

【図 2】

同実施形態に係る設計支援装置の実行制御表変換部の処理手順の一例を示すフローチャート

【図 3】

同実施形態に係る設計支援装置のシステム実装変換部の処理手順の一例を示すフローチャート

【図 4】

状態遷移表形式の仕様の一例を示す図

【図 5】

状態遷移表形式の仕様から実行制御表形式の仕様への変換について説明するための図

【図 6】

状態遷移表形式の仕様から実行制御表形式の仕様への変換について説明するための図

【図 7】

状態遷移表形式の仕様の一例を示す図

【図 8】

実行制御表形式の仕様の一例を示す図

【図 9】

実行制御表形式の仕様からシステム記述言語形式の仕様への変換について説明

するための図

【図 1 0】

システム記述言語形式の仕様の一例を示す図

【図 1 1】

同実施形態に係るシステム設計支援装置の構成例を示す

【符号の説明】

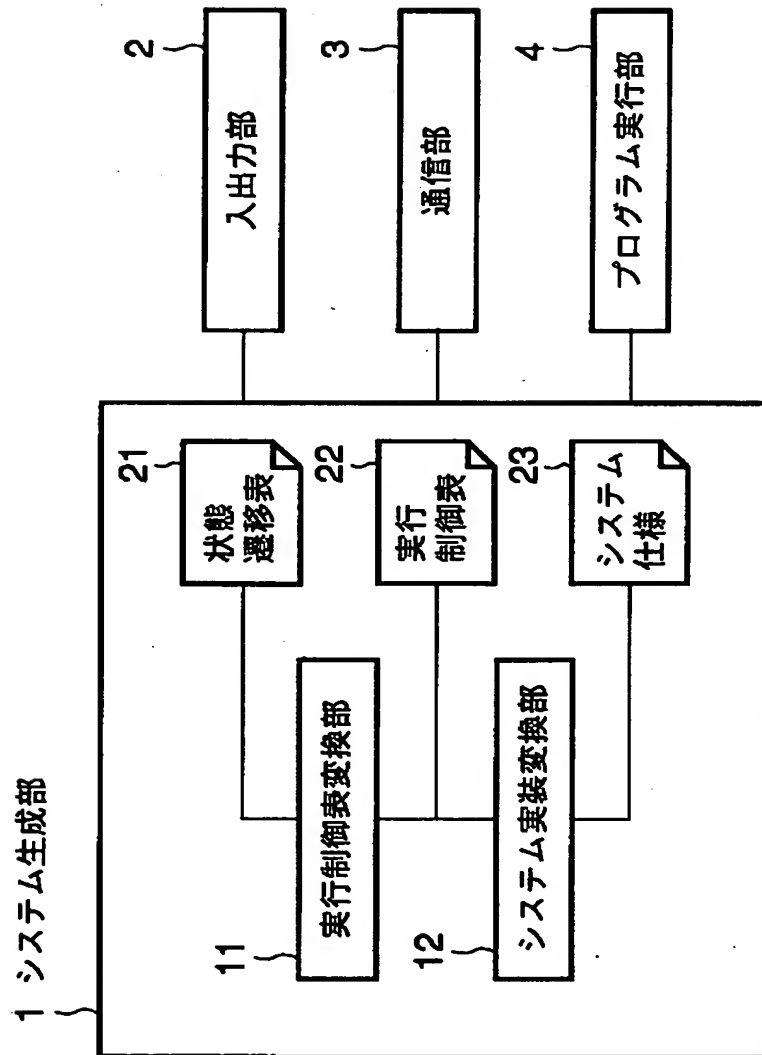
- 1 … システム生成部
- 2 … 入出力部
- 3 … 通信部
- 4 … プログラム実行部
- 1 1 … 実行制御表変換部
- 1 2 … システム実装変換部
- 1 0 0 … システム設計支援装置
- 1 0 1 … 仕様モデル記述部
- 1 0 2 … アーキテクチャ探索部
- 1 0 3 … コミュニケーション合成部
- 1 0 4 … ハードウェア仕様生成部
- 1 0 5 … 部品化・再利用部
- 1 0 6 … ソフトウェア仕様生成部
- 1 0 7 … システム仕様記録部



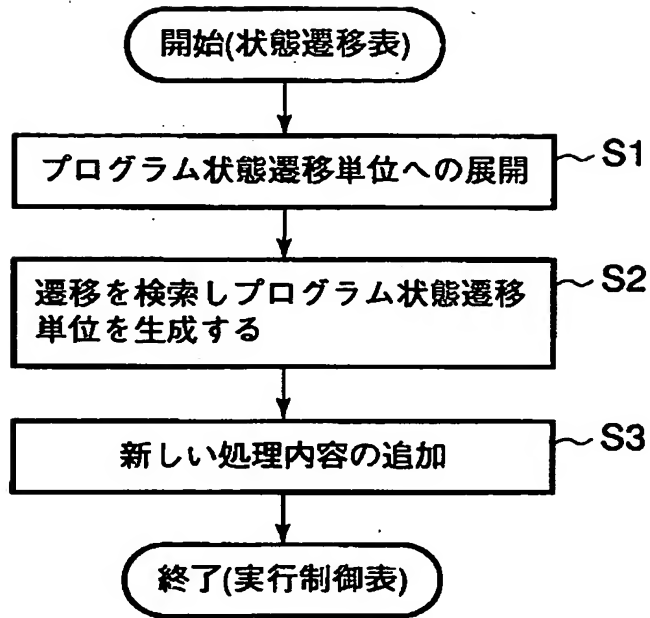
【書類名】

図面

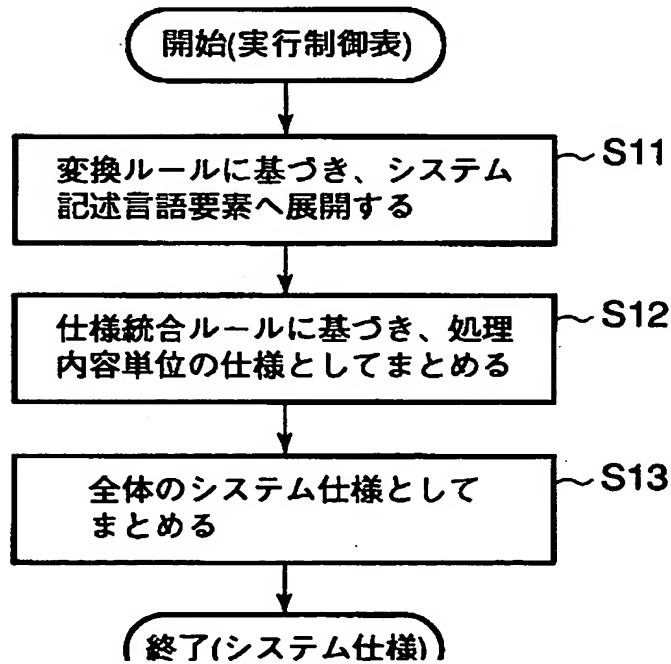
【図 1】



【図 2】



【図 3】



【図 4】

番号	現状態	イベント	条件	次状態	実行制御			
					start	stop	interrupt	resume finish
[ 0 ]	start			stopped	START			
[ 1 ]	stopped	<play>		playing	play			
[ 2 ]	playing	<pause>		pausing	pause		play	
[ 3 ]	playing	[FINISH]		stopped				play
[ 4 ]	pausing	<play>		playing	pause			play
[ 5 ]	pausing	<pause>		playing	pause			play
[ 6 ]	playing	<stop>		stopped	play			
[ 7 ]	pausing	<stop>		stopped	pause			

【図 5】

番号	現プログラム	遷移	起動タイプ	終了タイプ	次プログラム
	play	[2]	起動	割込一時停止	pause
	pause	[4]	復帰	割込終了	play
	pause	[5]	復帰	割込終了	play
	START	[0]		割込終了	
		[1]	起動		play
	play	[3]		完了	
	play	[6]		割込終了	
	pause	[7]		割込終了	

【図 6】

番号	現プログラム	遷移	起動タイプ	終了タイプ	次プログラム
	pause	[7]→[1]	起動	割込終了	play
	play	[6]→[1]	起動	割込終了	play
	play	[3]→[1]	起動	割込終了	play
	START	[0]→[1]	起動	完了	play

【図 7】

番号	現状態	イベント	条件	次状態	実行制御			
					start	stop	interrupt	resume finish
[0]	start			stopped	Xstop			
[1]	stopped	<play>		playing	play	Xstop		
[2]	playing	<pause>		pausing	pause		play	
[3]	playing	[FINISH]		stopped	Xstop			play
[4]	pausing	<play>		playing		pause		play
[5]	pausing	<pause>		playing		pause		play
[6]	playing	<stop>		stopped	Xstop	play		
[7]	pausing	<stop>		stopped	Xstop	pause		

【図 8】

番号	現プログラム	遷移	起動タイプ	終了タイプ	次プログラム
(0)	START	[0]	起動		Xstop
(1)	Xstop	[1]	起動	割込終了	play
(2)	play	[2]	起動	割込一時停止	pause
(3)	play	[3]	復帰	完了	Xstop
(4)	pause	[4]	復帰	割込終了	play
(5)	pause	[5]	復帰	割込終了	play
(6)	play	[6]	起動	割込終了	Xstop
(7)	pause	[7]	起動	割込終了	Xstop

【図 9】

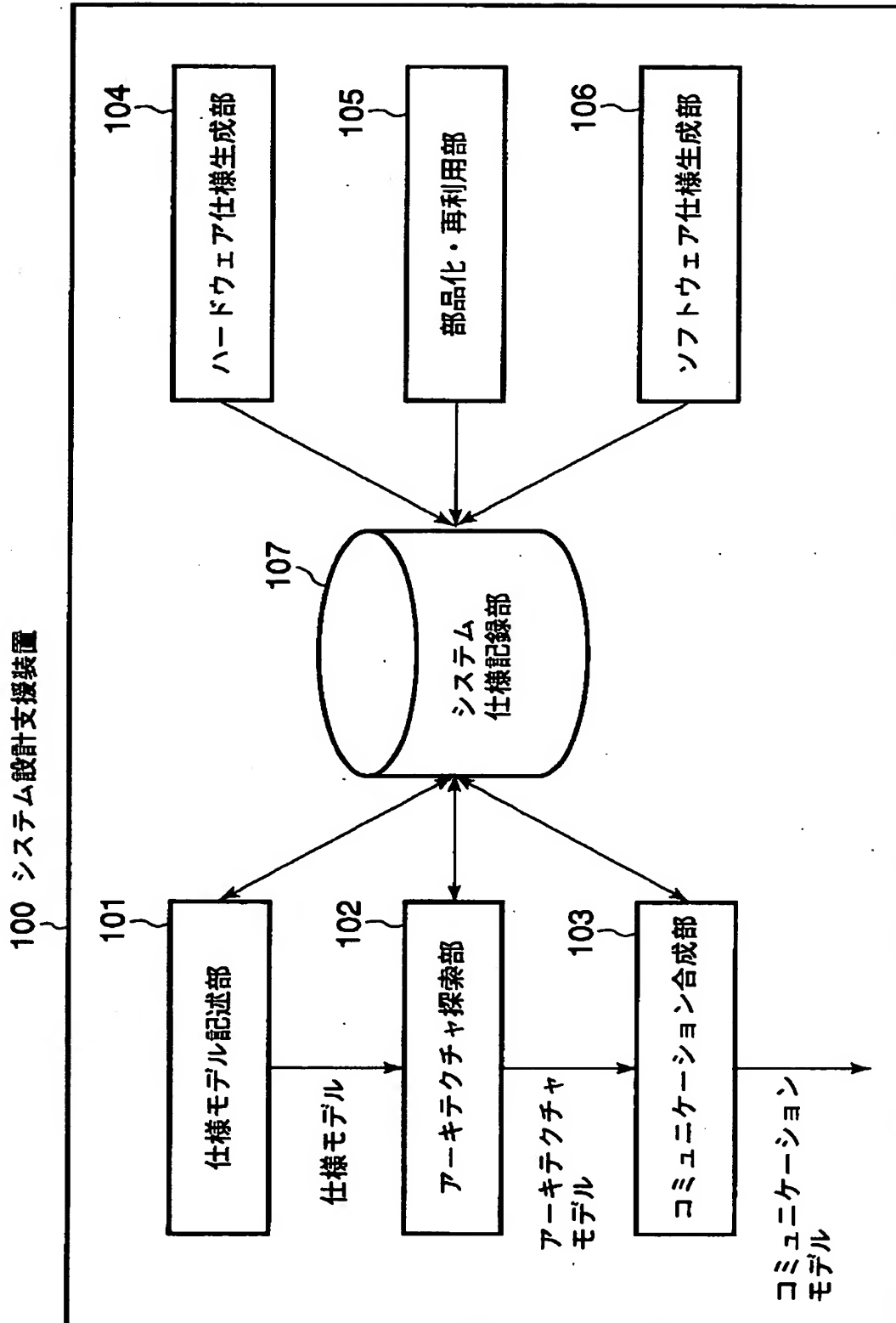
番号	現 フーズム	遷移	起動 タイ	終了タイ	次 フーズム	変換結果
(0)	START	[0]	起動		Xstop	fsm{wait(start), Xstop, goto(1)}
(1)	Xstop	[1]	起動	割込終了	play	fsm{wait(ev_Xstop), play, goto(1)}
(2)	play	[2]	起動	割込 一時停止	pause	try{Xstop}trap(ev_1){notify(ev_Xstop_stop)} fsm{wait(ev_play_interrupt), pause, goto(1)} try{play}itrap(ev_2){fsm{notify(ev_play_interrupt), wait(ev_play_resume)}} fsm{wait(ev_play_end), Xstop, goto(1)}
(3)	play	[3]	起動	完了	Xstop	fsm{play, notify(ev_play_end), goto(1)}
(4)	pause	[4]	復帰	割込終了	play	fsm{wait(ev_pause_stop_resume), notify(ev_play_resume), goto(1)}
(5)	pause	[5]	復帰	割込終了	play	try{pause}trap(ev_4){notify(ev_pause_stop_resume)} fsm{wait(ev_pause_stop), notify(ev_pause_resume), goto(1)}
(6)	play	[6]	起動	割込終了	Xstop	try{pause}trap(ev_5){notify(ev_pause_stop)} fsm{wait(ev_play_stop), Xstop, goto(1)}
(7)	pause	[7]	起動	割込終了	Xstop	try{play}trap(ev_6){notify(ev_play_stop)} fsm{wait(ev_pause_stop), Xstop, goto(1)} try{pause}trap(ev_7){notify(ev_pause_stop)}

【図 10】

```
par{
    fsm{wait(start, ev_play_end, ev_play_stop, ev_pause_stop),
        try(Xstop)trap(ev_1){notify(ev_Xstop_stop)},
        goto(1)
    },
    fsm{wait(ev_Xstop_stop),
        try{fsm{play, notify(FINISH)}}
        itrp(ev_2){fsm{notify(ev_play_interrupt), wait(ev_play_resume)}}
        trap(ev_6){notify(ev_play_stop)}
        trap(FINISH){notify(ev_play_end)},
        goto(1)
    },
    fsm{wait(ev_play_interrupt),
        try{pause}trap(ev_5){notify(ev_pause_stop_resume)}
        trap(ev_7){notify(ev_pause_stop)}
        trap(ev_4){notify(ev_pause_stop_resume)},
        goto(1)
    },
    fsm{wait(ev_pause_stop_resume), notify(ev_play_resume), goto(1)}
}
```



【図 11】



【書類名】 要約書

【要約】

【課題】 システム分析結果に対応した精度良いシステム設計の支援を行って開発期間を短縮し設計品質を向上できる設計支援装置を提供すること。

【解決手段】 状態遷移表形式のシステム仕様（21）を、現状態、イベント、次状態、実行制御に関する情報を含む状態遷移単位情報の集合で構成する。実行制御に関する情報は、当該状態遷移に関連して実行制御されるプログラム及びその実行制御内容（起動、割込終了、割込一時停止、終了、復帰）を含む。実行制御表変換部11は、この状態遷移表に基づいて、実行制御表形式のシステム仕様（22）を生成する。実行制御表は、遷移、現実行処理内容、その終了タイプ、次プログラム、その起動タイプを含むプログラム状態遷移単位情報の集合からなる。システム実装変換部12は、実行制御表に基づいて、システム記述言語で記述された実行可能な形式を持つシステム仕様（23）を生成する。

【選択図】 図1

出 願 人 履 歴 情 報

識別番号 [000003078]

1. 変更年月日 1990年 8月 22日  
[変更理由] 新規登録  
住 所 神奈川県川崎市幸区堀川町72番地  
氏 名 株式会社東芝
2. 変更年月日 2001年 7月 2日  
[変更理由] 住所変更  
住 所 東京都港区芝浦一丁目1番1号  
氏 名 株式会社東芝